A MIDDLEWARE FOR THE MANAGEMENT OF LARGE UTILITIES PLANTS

S. Cavalieri, F. D'Urso, C. Floridia, A.Rossettini
University of Catania, Faculty of Engineering, Department of Computer and Telecommunications Engineering, Viale A.Doria, 6 - 95125, Catania (ITALY)

Keywords: Middleware, Gas and water distribution utilities, Wastewater treatment systems, XML, Web Services, SCADA, GIS, DSS.

Abstract: The paper presents the main features of the European project Mobicossum IST 1999-57455, still running. The project is a CRAFT one approved inside the Fifth Framework Programme. It aims to define a middleware offering services for the management of large plants, in the field of gas and water distribution and wastewater treatment systems. In the paper, the main features of the project will be explained, focusing on the description of the implementation of the core of the middleware, called Generalised Interface.

1 INTRODUCTION

The Mobicossum project, IST 1999-57455, is a CRAFT project (V European Framework Programme) involving medium and small European enterprises working in the field of gas and water distribution and wastewater treatment. One of the RTD performers inside the project is the Department of Computer and Telecommunications Engineering at the University of Catania (Italy), to which the authors of the paper belong. The main objective of the Mobicossum project is to design and implement software technologies that allow the use of wireless mobile device as client of IT systems to control and manage large utilities plants. The main feature of the proposed technology is the definition of a common interface between mobile applications, running in mobile devices like palm PCs, and data stored in SCADA (Supervisory Control and Data Acquisition), GIS (Geographic Information System) and DSS (Decision Support System) applications. Common interface is not linked to a specific application, product or vendor. Further, platform independency is guaranteed. The common interface provides for several services useful in utilities management, like localisation, user authorisation, and business logic. The research focuses on satisfying requirements present in utilities for water and gas distribution and wastewater treatment systems, the three interest areas of the partners involved in the project. It must be clear that the results obtained can be easily transferred to other kind of utilities, e.g. electricity distribution. The common interface between mobile devices and applications like SCADA, GIS and DSS is realised defining a Middleware.

The paper will focus on the description of the Middleware from the implementation point of view, highlighting the software technologies adopted. In particular the description will be limited to the core of the Middleware called Generalised Interface.

2 MAIN FEATURES OF MOBICOSSUM PROJECT

Mobicossum aims to define a Middleware conceived to offer specialised services in the field of water/gas distribution and wastewater treatment systems. Services are mainly dedicated to mobile users (i.e. users connected to mobile devices, like mobile phones and PalmPCs). Middleware is placed between (mobile) users and the applications providing for information coming from water/gas distribution and wastewater treatment systems. These applications are SCADA, GIS and DSS, three kinds of applications featuring very different services and usage. Mobicossum is made up by the following subsystems: Presentation Manager (PM), Central Services (CS), Logical View (LV), Data Management (DM) and the Generalised Interface (GI).

Cavalieri S., D'Urso F., Floridia C. and Rossettini A. (2004).
A MIDDLEWARE FOR THE MANAGEMENT OF LARGE UTILITIES PLANTS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 99-104
DOI: 10.5220/0002601200990104

The Presentation Management is responsible to generate the pages presented to the users as a function of the user (name), the location of the user, the used presentation device (Pocket PC, WAP phone, etc.), the application status, etc.

The Central Services provide for different features. One of this is User Access Management, as access from not authorised users must be avoided, providing for secure access logging. Furthermore, when a particular user accesses the system by a secure logging, it is required that the middleware identifies its profile (e.g. technician, manager, and so on). In this way, the Mobicossum system may be automatically aware of the data needed by the user, avoiding the need to explicitly request the data desired. Other Central Services are those concerning Localisation. Localisation of each mobile worker is a very important requirement in large plant, like those related to water/gas distribution and wastewater treatment systems. Localisation is very important, for example, concerning alarm problems; a warning system, being aware of the location of each mobile worker, would allow mobile workers nearest to the warning area to be alarmed. Localisation may be used also for data selection based on the position of the user inside or outside the plant; only the data relevant to the geographical area nearest to the user are sent to him, avoiding transmission of useless information.

The Logical View provides for high-level services. It contains the business logic of Mobicossum application. The LV must be seen as a library of macro-level functions able to perform complex transformation on the real data available at the SCADA, GIS and DSS level. The type and complexity of the transformation depends on the kind of user calling the function.

The Data Management is in charge of providing data brokerage. It collects data from the different application connected to Mobicossum. The data can be retrieved in function of logical names or in function of a position specification.

The Generalised Interface is the core of the Middleware. This subsystem directly interfaces to the SCADA, GIS and DSS applications maintaining the real data needed by the Mobicossum clients. The main aim of the GI is that to offer a common interface to every SCADA, GIS and DSS application. This mainly means the definition of a unique set of services to access data maintained in SCADA, GIS and DSS applications and unique way to access these data. Decoupling with the real SCADA, GIS and DSS application is performed in this way. Services offered by the GI may be accessed directly by a user, but can also be used by the other components of Mobicossum Middleware, when they need to access the real data.

In order to understand the exchange of information between the components of Mobicossum, Figure 1 summarises its internal architecture. As can be seen, the GI is placed at the lowest level in order to provide for basic accesses to the SCADA, GIS and DSS applications. Exchange of information between Mobicossum components may occur from the top towards the bottom of the architecture (and vice versa) and inside each sub layer (between Data Management, Logical View and Central Services). A direct exchange of information between Presentation Management and the GI may be possible. Further, direct exchange of information between Mobicossum client and the GI is foreseen. In any case the access to the real data maintained to SCADA, GIS and DSS applications has no sense in Mobicossum, as all the query to these applications must be realised through the Generalised Interface.

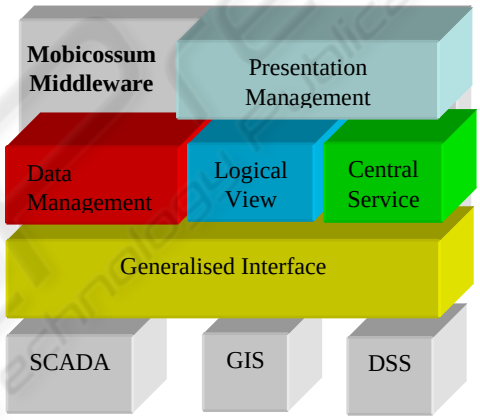

Figure 1: Mobicossum Middleware Architecture.

3 GENERALISED INTERFACE

Client applications interact with the Generalised Interface, which is in charge of receiving incoming messages containing requests for a particular service, parsing the messages, and dispatching the request to the appropriate method of a specific application such as a SCADA system, a GIS or DSS. When the service returns a response, the GI is also responsible for packaging the response into a message and sending back to the client. Definition of client applications must be clearly stated. Mobicossum project mainly focuses on definition of services (in the area of water/gas distribution and wastewater treatment systems) mainly available from mobile devices. In particular very simple mobile devices are considered, like mobile phones and PalmPCs. So client applications must be strictly

considered as very "simple" applications running on the mobile devices above mentioned. A browser running on a PalmPC must be considered an example of a very "simple" application running on a mobile device. However, the description of the implementation of the GI, contained in the following sections, will point out that extension (outside the aim of Mobicossum project) of client applications to more "intelligent" applications (let think, for instance, to a SCADA applications which has to collect information from other SCADA applications through Mobicossum infrastructure) is quite trivial.

Definition of the GI has been made taking into account two main requirements of control/monitoring problems in the area of water/gas distribution and wastewater treatment systems. The first concerns the Definition of Common Services. SCADA, GIS and DSS applications offer specialised services, whose meaning and scope is common to the most part of the products available in the market. But, their implementation is strictly linked to the particular SCADA, GIS and DSS application. Considering for example a SCADA application, it's always be possible to find groups of services conceived to retrieve and update information from/to field devices, but the names, the relevant syntax, the number of parameters, the error codes returned by the services are totally different changing from one SCADA application to the other. The main aim in the definition of the Generalised Interface is that to offer to client applications (including the other Mobicossum modules, seen as client for the GI) a unique set of services for each SCADA, GIS and DSS applications. Each service defined in the GI must be easily mapped into the real corresponding service offered by the particular SCADA, GIS and DSS application. The definition of the services for the GI has been based (in Mobicossum project) on a study carried on during the first months of the project. It pointed out the user requirements, in the sense of services real needed by users in the field of water/gas distribution and wastewater treatment systems.

The other requirement taken into account in the development of the GI is related to the Definition of a Common Access to the Services. One of the problems strongly felt when accessing to different SCADA, GIS and DSS applications, is linked to the different mechanisms used to achieve the access. Different solutions, like COM, DCOM, OPC currently exist and development of different accesses for different application is often required. In particular, development of a client application often means including different access schema according to the SCADA, GIS, DSS application the client application needs to access. Accessing to a new SCADA, GIS, DSS application implies to add new accessing schema for the specific application. For this reason, definition of Middleware and in particular definition of the GI has been performed with the aim of making platform and language independent the way of access to industrial applications by the client. The paper mainly focuses on this last aspect, i.e. the definition, inside Mobicossum project, of a common access to the services provided for by the GI.

4 GI ARCHITECTURE

The need of a common access mechanism to the services available at the SCADA, GIS and DSS level has led to the definition of an internal architecture of the GI featuring a unique access mechanism to all the applications below the GI, integrated into the Mobicossum environment.

Current literature presents several approaches for definition of communications between distributed applications, but that based on Web Services technology features a lot of advantages, due to the platform- and language-independence, and to the low configuration complexity and costs. For this reason, development of the GI was mainly based on Web Service technology. In particular it was assumed that the GI exports the own services using the Web Service technologies (Web Services, 2002). These services are those defined trying to generalise the services generally offered by SCADA, GIS and DSS applications available on the market. Due to the use of Web Service technology, these services have been defined in terms of Web Methods. Web Service technology has been used also for the exchange of information between GI and the SCADA, GIS and DSS applications. It is assumed that the only requirement for the integration of existing applications into Mobicossum environment is that these applications export their own functionalities through Web Services in order to establish the data exchange with Mobicossum Middleware.

XML/SOAP based communications has been assumed to send/receive request/response to/from the Generalised Interface and to realise the exchange of information between the Generalised Interface and the Web Services related to each application.

Figure 2 shows the integration between GI and the SCADA, GIS and DSS applications through the Web Services.

4.1 Integration with Existing Applications

As said before, the main hypothesis on the basis of Mobicossum Middleware is that integration of existing application is realised through Web Service-based technology. This means that each SCADA, GIS and DSS application has to provide for Web Services in order to be integrated into Mobicossum framework. Mobicossum Middleware, and in particular the GI, has to acquire a complete know-how about the applications below it and their available Web Services, including all the Web Methods offered. This is realised through the definition of a *Web Service Specification* for each application. A *Web Service specification* is an XML-Schema file (XML Schema, 2003) describing both Web Methods offered by Web Services related to each SCADA, GIS and DSS application, and the format of the XML/SOAP messages to be exchanged with these Web Services. According to the Web Service-based philosophy, also the Generalised Interface has to provide for a *Generalised Interface Specification*, describing the generalised services offered by the GI, using XML-Schema.

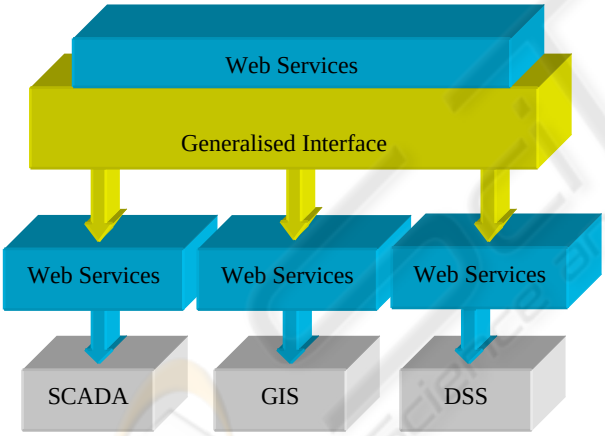

Figure 2: Integration between GI and SCADA, GIS and DSS Applications

On the basis of the Generalised Interface Specification and the Web Service Specification related to application registered inside Mobicossum, particular mapping files must be defined. In particular, two mapping files are needed for each application as they will be used by a particular component of the GI (the Mapper) to map the request/response written according to Generalised Interface Specification, into request/response written according to Web Service specification of the specific application, as will be explained in the following. It was assumed to realise the mapping of XML/SOAP document using XSLT language. So, suitable Mapping files written in XSLT language, are available to realise the translation, as already said before.

4.2 Main Components of the GI

The GI is made up by the following internal components: Parser, Mapper and Dispatcher, as shown by Figure 3. The Parser receives the XML\SOAP request coming from the client of the GI and checks if the XML document is valid. For the document validation, the Parser verifies that the XML/SOAP message is well formed and has been prepared according to an XML-Schema describing the correct structure of the incoming requests. The XML-Schema is the *Generalised Interface Specification*, as said before. If the request doesn't match, an error message will be returned; on the other hand, if the document is valid, the request will be sent to the Mapper module. For each response coming from the application, the Parser will receive a document from the Mapper component (described in the following). In this case the Parser will check that the format of the document has been prepared according to the XML-Schema for the response, passing the valid document to the client.

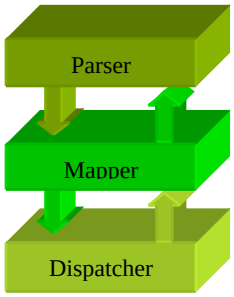

Figure 3: Generalised Interface Internal Architecture

The Mapper performs the translation of XML/SOAP request, sent to GI, into the format recognised by destination application. As said before, this mapping has been realised on the basis of suitable Mapping files written in XSLT language (XSLT, 2003). When the Mapper receives a request, written according to *Generalised Interface Specification*, it maps the request in the XML/SOAP request format of the destination application, using a Request Map file, that is the map file (in XSLT language) related to the specific application. The document resulting by mapping represents the format of the request that must be delivered to the specified destination application. Once the Mapper has processed a request, the request is passed to the

Dispatcher. The same process is made for the response, coming from the application. To map the response, coming from application, the Mapper uses another map file (Response Map file) that allows translating the message wrote according to the format of the specific application in the format defined by GI specification.

For each request forwarded by the Mapper, the Dispatcher makes the following steps. It finds the correct HTTP address of Web Service relevant to the requested application and it establishes the HTTP connection with web service. Then, it sends request and receives response. Finally, it handles the cookies, as explained in the following.

In principle, each instance of SCADA, GIS and DSS application may feature more than one Web Services, each of these exporting different Web Methods relevant to the same application. It's clear that client cannot know the address of each Web Service, but it can know the application instance. This information is specified by the client in a particular field, the ApplicationID, contained in a field of the XML/SOAP request sent to the GI. Moreover the client obviously can specify the method (expressed in terms of the *Generalised Interface Specification*) to be invoked. The service name is contained in the XML/SOAP request sent to the GI, too. Identification of the right Web Service address is performed by the Dispatcher on the basis of the ApplicationID and the GI service name requested by the client, using a database local to the Dispatcher. This database contains for each ApplicationID and for each GI Web Method, the corresponding Web Service Address. When the Dispatcher finds the right Web Service Address, it opens an HTTP connection to the correct web service. Then it performs the remote procedure call to the suitable web method and waits for the response. When the response arrives, the Dispatcher closes the connection and delivers the response to the Mapper.

A particular problem concerning the communication between Dispatcher and the Web Services of each application instance is that the HTTP is a stateless protocol. Each request to a Web Service is independent, and the application retains no memory of a client’s past requests. To overcome this limitation, management of the sessions state has been realised using HTTP cookies.

The idea behind HTTP cookies is that when the Dispatcher sends a request, the Web Service sends back a response with an HTTP Set-Cookie header that has a name/value pair in it. For all subsequent requests to the same Web Service, the Dispatcher sends the name/value pair in an HTTP Cookie header. The Web Service then can use the value to associate the subsequent requests with the initial request. The Web Service automatically processes this cookie and uses it to restore the values saved in the memory for this particular client. For each service requested by a client, the application will preserve the information stored for the client's session. This session information is stored in memory on the Web Service of the application. The client is provided with a unique cookie that the application uses to match client requests with the information specific to that client's session. A session state ends when the client does not make any HTTP requests to the application for a specified time-out period. When the Dispatcher sends the first request for a client, it receives the cookie from Web Service and stores it in a local database. For all the subsequent requests coming from the same client, the Dispatcher gets the cookie related to the client. Since the GI connects to several applications, it is necessary to keep the connection state for each client and for each application requested by the client.

4.3 Management of Multiple Connections

GI must be able to handle concurrent multiple connections concerning different requests by the same client or by different clients. For each request, it's required that an instance of the GI modules (Parser, Mapper, Dispatcher) is created. The instance must be deleted when the request has been satisfied.

5 GI IMPLEMENTATION

The architecture of the GI described in the previous section has been implemented in such a way to be not linked to a specific platform, but it has been realised using freeware libraries.

The implementation was realised on Windows 2000 Operating System. This is only an implementation choice, not existing constraints on the use of this OS, as it will be pointed out in the next.

An analysis of the state of the art about free software/libraries useful to manipulate XML files for the parser, mapper and dispatcher has been done, during the first stage of the project. It highlighted the advantages in using DOM libraries to realise parsing of XML documents and Dispatcher functionalities, and XSLT engine to realise the Mapper.

DOM presents an easy processed standardised interpretation of an XML document to applications and scripts. Different free implementations of the DOM exist in different languages. We have used the Microsoft XML Core Services (MSXML), realised as DLL implementing the parser DOM in a COM component (MSXML, 2003).

In order to implement the Mapper, use of the XSLT engine has been considered. The XSLT engine can be an external component, a library or a class. Different free implementations of this engine exist, among which the MSXML library used in the implementation (MSXML, 2003).

Dispatcher has been implemented as class. It is instanced after the Mapper has performed the SOAP request transformation. The Dispatcher exports one method that implements the routine to handle the communication with the Web Service of each specific SCADA, GIS and DSS application. Also for this implementation we have used the MSXML library; in particular we have used the ServerXMLHTTP interface. It provides methods and properties that enable establishment of an HTTP connection between files or objects on different Web servers (MSXML, 2003). The ServerXMLHTTP object is used to handle the HTTP protocol, such as to post SOAP documents to a remote Web Service (HTTP POST), to read and to insert cookie in the Header HTTP, read the response status code, etc (MSXML, 2003).

5.1 Software Requirements

Implementation based on the use of DOM and XSLT Engine is featured by very few constraints on the software requirement. DOM and XSLT Engine are not linked to a specific Operating System, as it's possible to find libraries for Windows and for Unix platform. Management of Web Services can be realised by IIS or by Apache for example, but there is no constraints on this item. From the implementation point of view, the GI has been developed assuming to use: Windows 2000 platform, Visual Studio .NET development environment. This has implied the following choices: a DOM library compliant with Windows 2000 and Visual Studio .NET (one of the libraries available on the web is the Microsoft MSXML library, that has been used in the implementation) and IIS for developing and support the Web Services. If the solution based on DOM and XSLT Engine isn't linked to a specific platform, on the other hand it requires particular tools (Editor and Mapper), and in same cases they are not free. Used tools in this implementation were: XML Spy5 Editor and TIBCO XMLTransform 1.1.0 (this last in order to produce map file).

5.2 Management of Multiple Connections

The implementation of the GI carried out is able to receive and process many requests at the same time. The Web Services exporting the functionalities of the GI are published in Internet through a web server. We have chose Internet Information Services (IIS) as web server. When a client of the GI calls a GI method, it sends an HTTP/SOAP request to the web server that publishes the web method. The web server receives the request and allocates a thread to process it. For each request, a thread is allocated, so the web server doesn't block and can accept other coming requests. In order to implement the web services, we have chosen the ASP.NET technology offered by the .NET Framework. In this case the web server allocates a thread of the .NET Run-Time to process the request. The thread loads and executes the class implementing the web service. In the execution of the web service, instances of the objects processing the SOAP request (Parser, Mapper and Dispatcher) are created. The Dispatcher sends the mapped request to the destination application web service. After, it waits for the web service response. When the response comes back, the Dispatcher forwards it to the client. At this point, the request of the client is completely satisfied, so the thread is destroyed together the instances of Parser, Mapper and Dispatcher.

6 CONCLUSIONS

The paper has presented an overview on the main features of the CRAFT IST 1999-57455 project, highlighting the internal architecture and implementation of the GI, which is the component in charge to offer to clients a set of services not linked to a specific product and implementation. The choice of a platform/language technology allowed overcoming the main limit of the current SCADA/GIS/DSS applications currently available, making vertical and horizontal integration of those applications possible. Vertical integration means that a generic client application is able to access any information maintained in a particular SCADA, GIS and/or DSS application without be linked to the access mechanism and syntax of the particular application. Horizontal integration means that a client application is allowed to acquire information coming from different SCADA/GIS and DSS applications.

REFERENCES

Web Service, 2002, http://www.w3.org/2002/ws/

XML-Schema, 2003, http://www.w3.org/XML/Schema

XSLT, 2003, http://www.w3.org/Style/XSL/

MSXML,2003,http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/htm/sdk_intro_6g53.asp